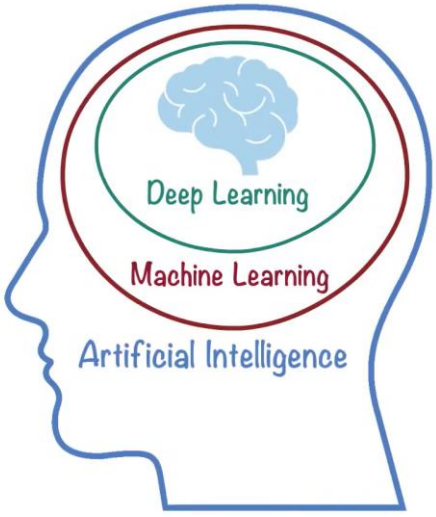
MODUL PRAKTIKUM PEMBELAJARAN MESIN

Pertemuan 6

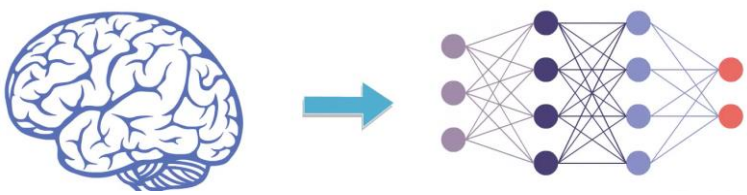
Pengenalan Deep learning

A. Pengenalan Deep Learning

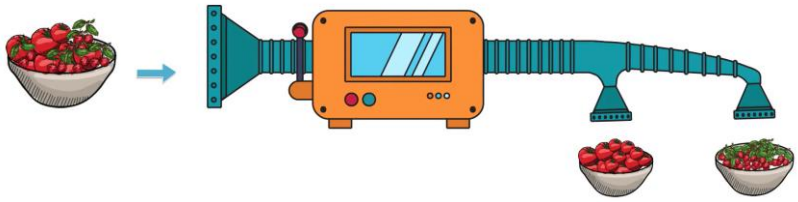
Deep learning merupakan subset dari *machine learning*, yang mana *machine learning* merupakan subset dari *artificial intelligence*.



Artificial intelligence (AI) adalah sebuah teknik yang dapat memungkinkan komputer meniru perilaku/ tindakan manusia. Kemudian *machine learning* adalah sebuah teknik untuk mencapai AI melalui algoritma yang dilatih dengan data. Sedangkan, *deep learning* merupakan tipe dari *machine learning* yang terinspirasi dari struktur dari otak manusia dan bagaimana cara otak manusia bekerja, yang mana dalam *deep learning* struktur ini disebut dengan *artificial neural networks*. Jadi, bisa dikatakan bahwa semua implementasi *deep learning* merupakan *machine learning*, tetapi tidak semua implementasi *machine learning* merupakan *deep learning*.



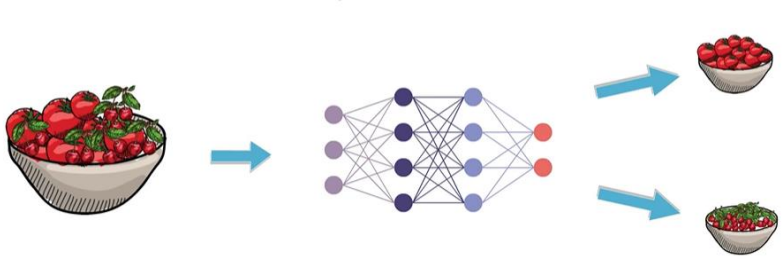
Secara sederhana, *machine learning* menggunakan algoritma untuk mengurai data, belajar dari data tersebut, dan membuat keputusan berdasarkan apa yang telah dipelajarinya. Namun, komputer masih berpikir dan bertindak seperti mesin. Kemampuan *machine learning* untuk melakukan beberapa tugas kompleks masih jauh dari kemampuan manusia. Sedangkan *deep learning* menyusun algoritma berlapis-lapis untuk menciptakan jaringan saraf tiruan (*artificial neural network*) yang mempelajari dan membuat keputusan sendiri. Sebagai contoh, misalnya kita membangun sebuah mesin yang dapat membedakan tomat dan ceri.



Jika kita membangun mesin ini dengan *machine learning*, kita (manusia) harus membuat data mengenai fitur-fitur dari tomat dan ceri, yang selanjutnya data ini akan dipelajari menggunakan algoritma *machine learning*, sehingga nantinya komputer dapat membedakan antara tomat dan ceri.

Fitur-fitur		
	Tomat	Ceri
Diameter	4-15 cm	2 cm
Rasa	Asam	Manis
dll		

Sedangkan, jika mesin tersebut dikerjakan dengan *deep learning*, maka fitur-fitur (yang nantinya akan membedakan) tomat dan ceri akan ditentukan oleh *neural nerworks* tanpa campur tangan manusia.



Tentu saja untuk meraih ini, diperlukan data dengan volume yang lebih besar untuk dilatih oleh mesin.

B. 'Hello World' terhadap Neural Networks

Sebelumnya, kita telah memahami bagaimana konsep dari *deep learning*. Selanjutnya mari kita lihat bagaimana *deep learning* bekerja. Misalnya terdapat kumpulan angka, seperti di bawah ini:

$$x = 0, 1, 2, 3, 4, 5$$
$$y = 1, 3, 5, 7, 9, 11$$

Coba tentukan formula yang dapat memetakan x ke y! Bagaimana Anda dapat menemukan formulanya? Mungkin Anda melihat bahwa setiap y bertambah 2, maka x bertambah 1. Jadi mungkin formulanya seperti y = 2x tambah sesuatu. Kemudian, Anda melihat bahwa ketika x = 0, y = 1, jadi Anda berpikir bahwa ada sesuatu yang ditambah 1, jadi mungkin formulanya y = 2x + 1. Anda mungkin mencoba beberapa nilai yang lain dan melihat apakah nilainya cocok. Jika Anda melakukan hal-hal di atas, selamat Anda telah melakukan dasar *machine learning* di kepala Anda sendiri!

Selanjutnya, mari kita lihat kode di bawah ini.

```
model = keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
```

Kode di atas ditulis menggunakan Python, TensorFlow, dan sebuah API dalam TensorFlow yang disebut dengan keras. Secara sederhana, *neural network* merupakan seperangkat fungsi yang dapat mempelajari pola. Dengan

keras, kita dapat mendefinisikan *neural networks* dengan sangat mudah. Dalam keras, kita menggunakan kata '**dense**' untuk mendefinisikan satu lapis neuron yang terhubung (*layer*). Menurut kode di atas, maka hanya terdapat 1 *layer neural network*, dan karena hanya terdapat 1 unit di dalamnya, maka hanya ada 1 neuron. Kita akan membangun sebuah model *layer-by-layer* dan didefinisikan secara berurutan, oleh karena itu kata **Sequential** digunakan. Tetapi, dalam kasus ini, kita hanya membuat 1 neuron. Selanjutnya, kita menentukan bentuk *input* yang akan dimasukkan ke jaringan saraf yang pertama, dan dapat dilihat bahwa *input* yang kita miliki sangat sederhana, yaitu hanya terdiri atas 1 nilai.

Perlu diketahui bahwa dalam *machine learning*, Anda perlu mengetahui dan menggunakan banyak sekali rumus matematika, probabilitas kalkulus, dan sebagainya. Sedangkan, dengan TensorFlow dan keras sebagian besar dari matematika itu telah diimplementasikan untuk Anda di dalam fungsi-fungsi yang telah disediakan. Namun, terdapat 2 fungsi yang harus Anda ketahui, yaitu *loss function* dan *optimizers*, kode di bawah ini akan mendefinisikannya:

```
model = keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')
```

Untuk memahami mengenai *optimizer* dan *loss function*, pikirkan seperti ini. *Neural networks* tidak mengetahui hubungan antara x dan y, jadi ia membuat dugaan/ tebakan. Katakanlah tebakannya y = 10x + 10. *Neural network* kemudian akan menggunakan data yang diketahuinya, yaitu data x dan y yang kita definisikan sebelumnya, kemudian akan mengukur seberapa baik atau seberapa buruk tebakan yang telah ia buat. *Loss function* akan mengukur hal ini, kemudian akan memberikan data ke *optimizer*, sehingga *optimizer* dapat mencari tahu tebakan berikutnya. Jadi, *optimizer* akan berpikir mengenai seberapa baik atau seberapa buruk tebakan yang sudah dilakukan dengan menggunakan data dari *loss function*. Maka, logikanya adalah bahwa setiap tebakan harus lebih baik daripada tebakan yang sebelumnya. Seiring tebakan menjadi lebih baik dan lebih baik lagi, akurasi akan mendekati 100 persen,

dalam hal ini istilah konvergensi digunakan. Dalam kasus ini, *loss function* yang digunakan adalah *mean squared error* dan *optimizer* yang digunakan adalah SGD, yaitu *stochastic gradient descent*.

Selanjutnya yang akan kita lakukan adalah merepresentasikan data yang diketahui. Kita dapat mendefinisikan data x dan y yang akan dilatih oleh model yang dibuat dengan kode di bawah ini:

```
model = keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

xs = np.array([0, 1, 2, 3, 4, 5], dtype=float)
ys = np.array([1, 3, 5, 7, 9, 11], dtype=float)
```

Np.array menggunakan *library* Python yang disebut dengan numpy, yang mana akan membuat representasi data, khususnya *list* jauh lebih mudah. Selanjutnya, kita dapat melatih data (*training*) dengan perintah **fit**.

```
model = keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

xs = np.array([0, 1, 2, 3, 4, 5], dtype=float)
ys = np.array([1, 3, 5, 7, 9, 11], dtype=float)

model.fit(xs, ys, epochs=500)
```

Melalui kode di atas, kita meminta model untuk mencari tahu bagaimana menyesuaikan nilai x dengan nilai y. Epochs = 500 memiliki arti bahwa model ini akan melakukan *training loop* sebanyak 500 kali. *Training loop* ini adalah apa yang sudah dijelaskan sebelumnya, yaitu membuat tebakan, mengukur seberapa baik atau seberapa buruk tebakan yang dibuat dengan *loss function*, kemudian menggunakan *optimizer* dan data untuk membuat tebakan selanjutnya, dan mengulangi semua proses yang sudah disebutkan tersebut lagi.

Ketika model telah selesai melatih data, kemudian model akan memberikan nilai tebakan terhadap suatu nilai dengan *method* ***predict***.

```
model = keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

xs = np.array([0, 1, 2, 3, 4, 5], dtype=float)
ys = np.array([1, 3, 5, 7, 9, 11], dtype=float)

model.fit(xs, ys, epochs=500)

print(model.predict([10.0]))
```

Nilai 10 tidak ada dalam data x yang sudah didefinisikan sebelumnya, jadi menurut Anda nilai apa yang akan dikembalikan model ketika ia menerima nilai 10? Ketika di-*run*, model akan memberikan nilai yang sangat dekat dengan 21, tetapi tidak tepat 21. Hal ini terjadi karena model dilatih dengan data yang sangat sedikit, yaitu hanya 5 poin data. Kelima data tersebut linier, tetapi tidak ada jaminan bahwa untuk setiap x, hubungannya akan menjadi y = 2x + 1. Ada kemungkinan yang sangat tinggi bahwa dengan x = 10, maka y = 21, tetapi jaringan saraf tidak positif, jadi jaringan saraf akan mencari nilai yang realistis untuk y. Selain itu, ketika kita menggunakan *neural networks* (jaringan saraf), seiring mereka mencoba untuk mencari tahu jawabannya, mereka akan berhadapan dengan probabilitas.

C. Pengenalan ke Computer Vision

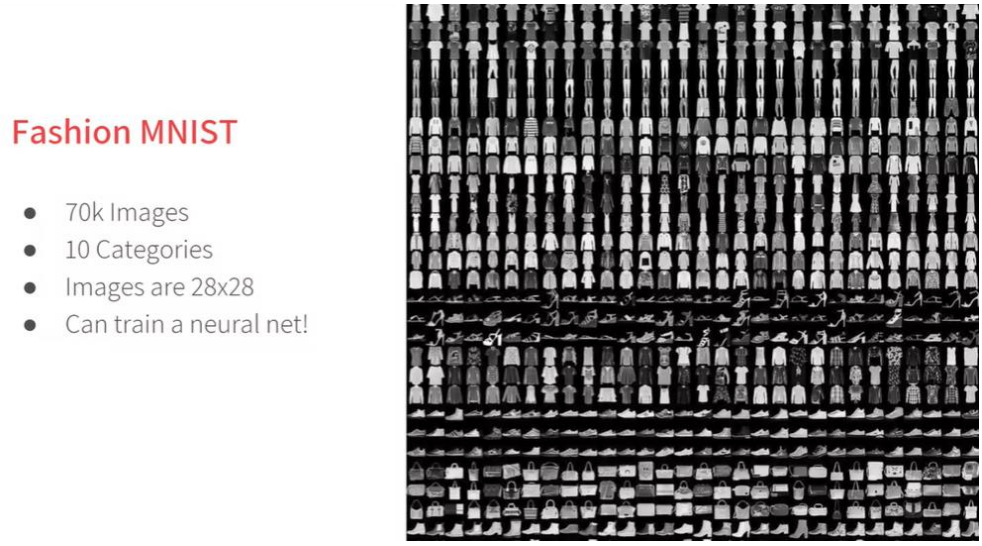
Pada poin sebelumnya, kita sudah melihat dan memahami konsep dasar dari *deep learning* dan mengimplentasikannya dalam kode menggunakan Python dan TensorFlow. *Machine learning* membuka banyak skenario baru yang dapat dilakukan, misalnya *activity recognition*, *object detection*, dan lain-lain. Pada poin sebelumya, kita telah menggunakan *machine learning* untuk menemukan hubungan antara x dan y. Tetapi, inti dari gagasan tersebut adalah dengan *machine learning*, kita dapat melakukan hal-hal menakjubkan lainnya

seperti, meminta komputer melihat gambar dan melakukan *activity recognition*, atau melihat gambar dan memberi tahu kita, apakah gambar tersebut adalah gaun, atau celana, atau sepasang sepatu. Manusia dapat dengan mudah mengatakan seperti apa wujud gaun atau seperti apa wujud sepatu. Tetapi bagaimana *programmer* dapat menulis aturan untuk itu dan memberikannya ke komputer, sehingga nantinya komputer juga dapat mengenali mana yang sepatu, mana yang gaun, mana yang celana, atau hal lain? Bagaimana cara *programmer* mengatakan bahwa jika pixel ini maka itu adalah sepatu, jika pixel itu maka itu adalah gaun. Hal ini sangat sulit dilakukan, jadi menggunakan *sample* berlabel adalah solusinya.

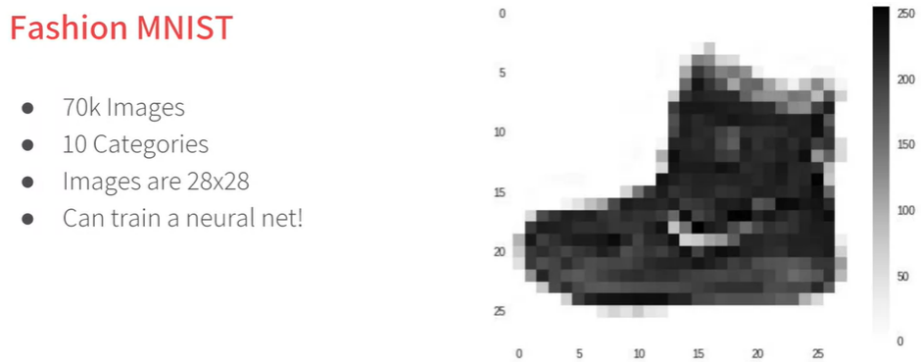
Jadi apa itu *computer vision*? *Computer vision* adalah sebuah bidang yang dapat membuat komputer memahami dan memberi label terhadap apa yang ada dalam sebuah gambar. Lihat gambar-gambar di bawah ini!

Ketika Anda melihatnya, Anda dapat mengenali yang mana yang merupakan kemeja, atau yang mana yang merupakan sepatu, yang mana yang merupakan tas, tetapi bagaimana Anda memprogramnya? Sangat sulit untuk menjelaskan kepada komputer bagaimana bentuk sepatu itu, atau bagaimana bentuk kemeja itu, masalah ini merupakan masalah yang sama pada *computer vision*. Jadi, salah satu cara untuk mengatasinya adalah dengan menggunakan sejumlah besar gambar pakaian dan memberi tahu komputer gambar apakah itu, kemudian meminta komputer mencari tahu polanya, dan mencari tahu perbedaan antara sepatu, kemeja, tas, dan mantel. Selanjutnya Anda akan diajak untuk belajar bagaimana cara melakukan hal tersebut menggunakan

dataset *Fashion* MNIST yang terdiri atas 70.000 gambar yang tersebar di 10 *item* pakaian yang berbeda-beda. Gambar-gambar ini telah diperkecil menjadi 28x28 pixel.



Setiap pixel dapat direpresentasikan dalam nilai dari 0 hingga 255, sehingga hanya ada 1 byte per pixel. Dengan 28 x 28 pixel dalam sebuah gambar, maka hanya diperlukan 784 byte untuk menyimpan seluruh gambar. Meskipun demikian, kita masih dapat melihat apa yang ada di dalam gambar, yang mana dalam hal ini adalah sebuah *boot*.



Dataset *Fashion* MNIST tersedia dengan panggilan API dalam TensorFlow. Berikut kode untuk mendeklarasikan objek tipe MNIST dan memuatnya dari database keras:

```
fashion_mnist = tf.keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

Dalam dataset *fashion* MNIST, 60.000 dari 70.000 gambar akan digunakan untuk melatih jaringan, kemudian 10.000 gambar lain yang belum pernah dilihat sebelumnya, dapat digunakan untuk menguji seberapa baik atau seberapa buruk kinerja jaringan. *Method* load_data akan mengembalikan 4 list kepada kita, yaitu data *training* berupa gambar, label-label dari data *training* yang berisi gambar apa itu sebenarnya, data *testing* berupa gambar, dan label-label dari data *testing* yang berisi gambar apa itu sebenarnya. Jadi, kode di atas akan memberikan kita 4 list tersebut.

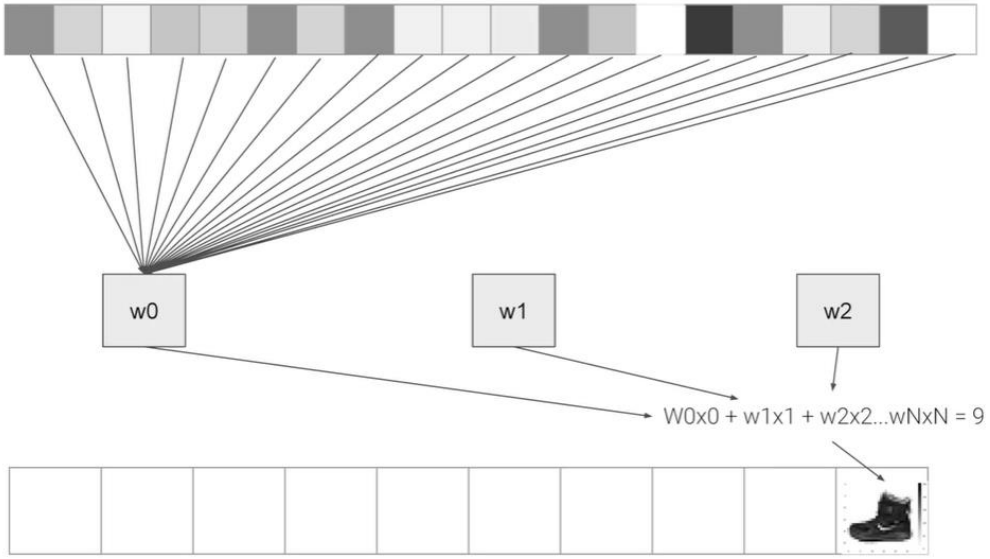
Label yang mendeskripsikan suatu gambar akan direpresentasikan dengan angka. Sebagai contoh, sesuai gambar di atas, *boot* direpresentasikan dengan nomor 9 karena komputer akan lebih memahami angka daripada teks. Hal ini juga mencegah terjadinya bias terhadap bahasa tertentu.

Selanjutnya kita akan mendefinisikan *neural networks* untuk melatih data *fashion* MNIST, sebagai berikut:

```
model = keras.Sequential([
        keras.layers.Flatten(input_shape=(28, 28)),
        keras.layers.Dense(128, activation=tf.nn.relu).
        keras.layers.Dense(10, activation=tf.nn.softmax)
```

])

Sekarang kita mendefinisikan 3 *layer* jaringan saraf. Hal penting yang perlu diperhatikan adalah *layer* pertama dan terakhir. *Layer* terakhir memiliki 10 neuron di dalamnya karena dataset *fashion* MNIST memiliki 10 kelas (*t-shirt*, *trouser*, *pullover*, *dress*, *coat*, *sandal*, *shirt*, *sneaker*, *bag*, dan *ankle boot*). *Layer* pertama adalah sebuah *flatten layer* dengan *input shape* 28 x 28. Jika Anda ingat, gambar kita berukuran 28 x 18, jadi di sini kita akan menspesifikasikan bahwa bentuk data yang akan masuk sebesar 28 x 28. *flatten layer* akan mengambil persegi dengan ukuran 28 x 28 ini dan mengubahnya menjadi array linier sederhana. Lapisan tengah biasa disebut dengan *hidden layer*/ lapisan tersembunyi yang mana terdiri atas 128 neuron.



Referensi :

(1) https://www.zendesk.com/blog/machine-learning-and-deep-learning/

(2) https://www.youtube.com/watch?v=6M5VXKLf4D4

(3) https://www.coursera.org/learn/introduction-tensorflow/home/week/1